```
class Program
{
   static void Main()
   {
      // Crear un objeto que deseas serializar en JSON
      var myObject = new { Name = "John", Age = 30, City = "New York" };
      // Convertir el objeto a JSON
      string jsonString = JsonSerializer.Serialize(myObject, new JsonSerializerOptions
      {
         WriteIndented = true
      });
      // Escribir el JSON en un archivo
      File.WriteAllText("datos.json", jsonString);
      // Leer el archivo JSON
      jsonString = File.ReadAllText("datos.json");
      // Convertir el JSON a un objeto
      var deserializedObject = JsonSerializer.Deserialize<JsonDocument>(jsonString);
      // Acceder a los datos del objeto deserializado
      string name = deserializedObject.RootElement.GetProperty("Name").GetString();
      int age = deserializedObject.RootElement.GetProperty("Age").GetInt32();
      string city = deserializedObject.RootElement.GetProperty("City").GetString();
      // Imprimir los valores en la consola
      Console.WriteLine($"Nombre: {name}");
      Console.WriteLine($"Edad: {age}");
      Console.WriteLine($"Ciudad: {city}");
   }
}
```

En este ejemplo, utilizamos la clase JsonSerializer para convertir el objeto myObject a una cadena JSON utilizando el método Serialize. También especificamos las opciones de serialización, como WriteIndented = true, para que el JSON se formatee de manera legible.

Luego, escribimos la cadena JSON en un archivo llamado "datos.json" utilizando File.WriteAllText.

Después, leemos el contenido del archivo JSON utilizando File.ReadAllText y lo almacenamos en la cadena jsonString.

Utilizamos la clase JsonSerializer nuevamente para deserializar la cadena JSON en un objeto de tipo JsonDocument utilizando el método Deserialize.

Finalmente, accedemos a las propiedades del objeto deserializado utilizando el método GetProperty de la propiedad RootElement. En este caso, obtenemos los valores de las propiedades "Name", "Age" y "City" y los imprimimos en la consola.

Trabajando con listas de json

Para guardar una lista en un archivo JSON y luego leerla en la clase principal de C# sin utilizar extensiones de terceros, puedes hacer uso de las clases y métodos proporcionados por el espacio de nombres System.Text.Json en .NET.

Aquí tienes un ejemplo de cómo lograrlo:

csharp

Copy code

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text.Json;

class Program
{
   static void Main()
   {
      // Crear una lista de personas
      List<Person> people = new List<Person>
```

```
        {
            new Person { Name = "John", Age = 30 },
            new Person { Name = "Alice", Age = 25 },
            new Person { Name = "Bob", Age = 40 }
        };

        // Convertir la lista a formato JSON
        string jsonString = JsonSerializer.Serialize(people);

        // Guardar el JSON en un archivo
        File.WriteAllText("people.json", jsonString);

        // Leer el archivo JSON
        jsonString = File.ReadAllText("people.json");

        // Convertir el JSON a una lista de personas
        List<Person> loadedPeople = JsonSerializer.Deserialize<List<Person>>(jsonString);

        // Mostrar los datos de las personas
        foreach (Person person in loadedPeople)
        {
            Console.WriteLine($"Nombre: {person.Name}, Edad: {person.Age}");
        }
    }
}

public class Person
{
    public string Name { get; set; }
```

```
    public int Age { get; set; }
}
```

En este ejemplo, creamos una lista de objetos Person y luego utilizamos JsonSerializer.Serialize para convertir la lista en una cadena JSON.

A continuación, utilizamos File.WriteAllText para guardar el JSON en un archivo llamado "people.json".

Luego, leemos el contenido del archivo JSON utilizando File.ReadAllText y lo almacenamos en la cadena jsonString.

Finalmente, utilizamos JsonSerializer.Deserialize para convertir la cadena JSON en una lista de objetos Person y mostramos los datos de las personas en la consola.

Ejemplos de json

```
// Root myDeserializedClass = JsonSerializer.Deserialize<List<Root>>(myJsonResponse);
    public class Root
    {
        [JsonPropertyName("name")]
        public string name { get; set; }

        [JsonPropertyName("role")]
        public string role { get; set; }

        [JsonPropertyName("lore")]
        public string lore { get; set; }

        [JsonPropertyName("abilities")]
        public List<string> abilities { get; set; }
    }
```

Archivo Json


```
[
 {
  "name": "Ahri",
  "role": "Mage",
  "lore": "Ahri is a nine-tailed fox who possesses the power to absorb the life essence of her enemies.",
  "abilities": [
   "Orb of Deception",
   "Fox-Fire",
   "Charm",
   "Spirit Rush"
  ]
 },
 {
  "name": "Ezreal",
  "role": "Marksman",
  "lore": "Ezreal is a charismatic explorer with the ability to shoot powerful energy bolts.",
  "abilities": [
   "Mystic Shot",
   "Essence Flux",
   "Arcane Shift",
   "Trueshot Barrage"
  ]
 },
 {
  "name": "Garen",
  "role": "Tank",
  "lore": "Garen is a noble warrior known for his unmatched strength and dedication to justice.",
```

```
    "abilities": [
      "Decisive Strike",
      "Courage",
      "Judgment",
      "Demacian Justice"
    ]
  }
]
```